ThePoorEngineer
WORKING HARD TO BE LAZY

MENU

MOTOR SPEED CONTROL

Posted on March 4, 2018

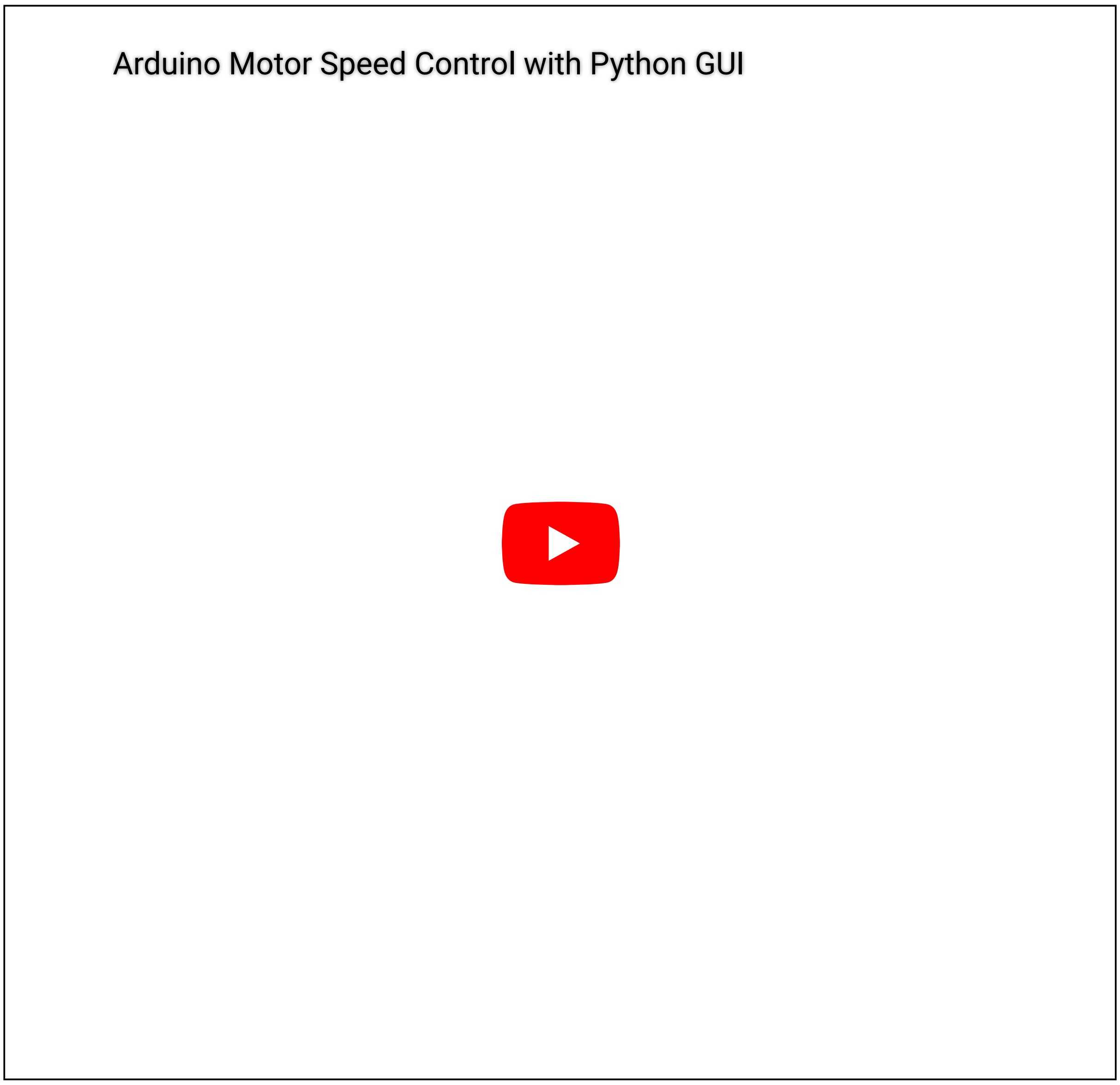
Table of Contents

In this chapter here, we will finally go into the actual implementation of a proportional controller. I will explain about the proportional controller in the coming sections so don't worry if you do not understand it yet at this point. In order to do this tutorial, you will need to set up the circuit based on my **previous post**. It may look complicated at first but it is actually quite straightforward.

Once you have built the circuit, you will need an interface to visualize the data measured by the Arduino. For this purpose, we will use Python to plot a real time graph, and also to communicate with the Arduino. You can get the source code for this part of the project with the link below but you may need to make some minute amendments based on the platform that you are running it with. If you are unsure, feel free to ask in the comments section or send me an email anytime.

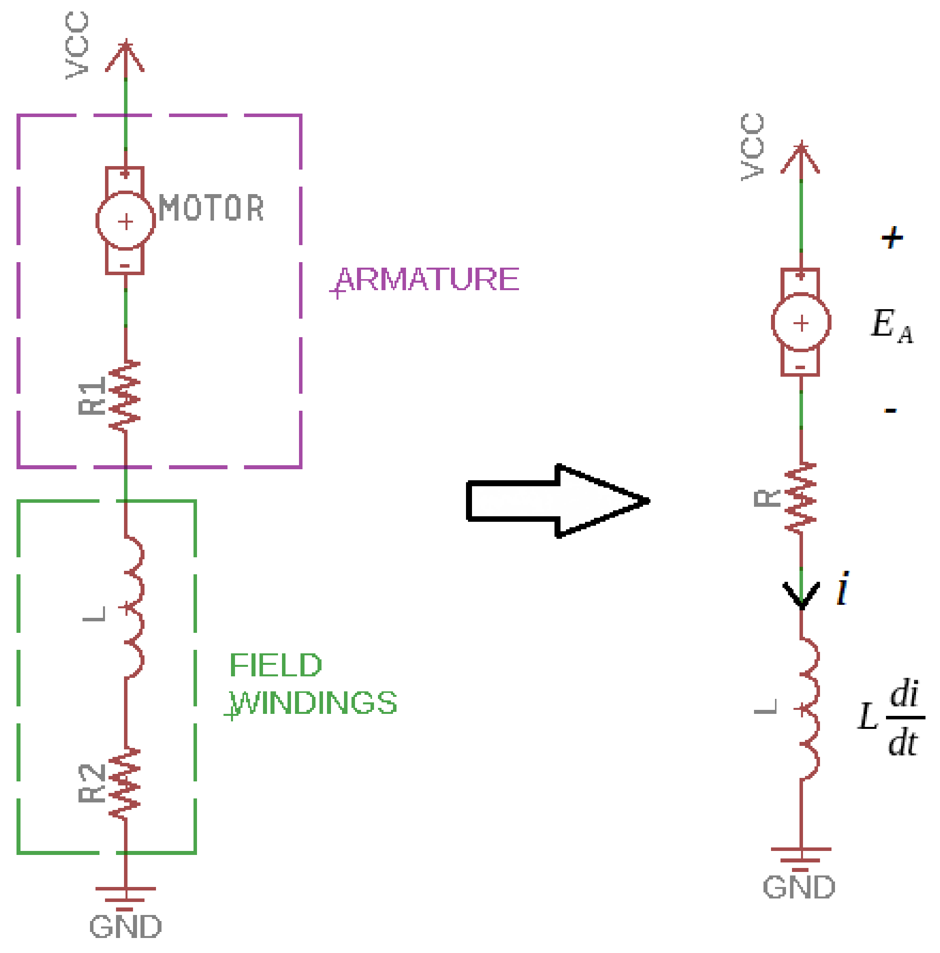
- **Python GUI Source Code for Arduino Motor Control**
- **Arduino Source Code for Motor Control**

In case you have not used Python before, you can follow **this post** to install a Python IDE to run the code. If you are looking to understand the code, you should read the previous 2 chapters on how to **plot Serial data** and **creating a GUI** with Python. Here's the final result first to catch your interest 🙂



Mathematical Model

In order to create a feedback control system, we will first need to have a model so that we can understand and predict the system's behavior. We will be using a DC series motor for this tutorial, and it simply means that the motor armature is connected in series with the field winding. The are many other type of motors around such as the DC Shunt motor (armature is connected in parallel to the field winding) and each of them have a different system characteristic. Below is a schematic diagram of the components in a DC series motor.



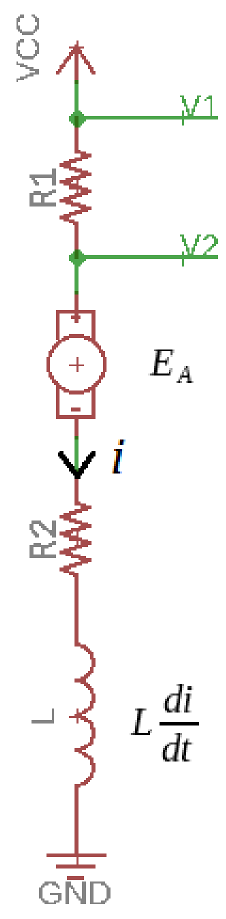
MOTOR COMPONENTS

Since we are only interested in the motor as a whole, resistance R1 and R2 can be combined to give a simpler representation as shown on the diagram on the right. Motors actually produce an electromotive force (EMF) when spun. You can easily ascertain this by connecting a voltmeter to your motor, and then by spinning the motor, the voltmeter should get a reading.

Whenever the motor spins, it will produce a back-EMF across its terminal that is proportional to the velocity of the motor. To be precise, it is only proportional over an interval, after which the back-EMF saturates. In this tutorial, we will be using this back-EMF as an estimate for the speed of the motor. The feedback control system will use the back-EMF as a reference but of course, you can use the velocity measured with an encoder as well.

The above diagram is actually the schematic for the motor alone. The red wire of the motor corresponds to the V_{CC} and the black wire corresponds to the GND in the schematic. This means that we cannot measure E_A directly. During steady-state, the current will be constant so $\frac{di}{dt} = 0$. Hence, the voltage across the inductor will be 0. However, the voltage across the resistor is not 0 so if we simply measure the voltage across the whole physical motor, we are actually measuring the back-EMF (E_A) together with the voltage across the motor intern resistor.

There is actually a very smart way to get around this. Take a look at the diagram below, which is actually the **circuit** that we are using.



MOTOR CIRCUIT

By adding another resistor before the motor and reading the voltage before and after the resistor, it is possible to determine the current in the circuit.

$$i = \frac{V1 - V2}{R1}$$

Thereafter, if we know the internal resistor, R2, it will be possible to calculate the back-EMF of the motor indirectly. Is it possible to determine the internal resistance of the motor? The answer is Yes! I will talk about this in the calibration section below so lets just keep this at the back of our head for now.

Assuming that we have obtained the current in the circuit, we can proceed to calculate the back-EMF as follows:

$$V2 = E_A + i(R2)$$

$$E_A = V2 - i(R2) \text{ ————- } (1)$$

Notice that we have ignored the inductor here. This is because we are assuming that the response of the inductor is fast, thus it will already be in steady-state when we do our measurements. During steady-state, the current across the inductor is constant so $\frac{di}{dt} = 0$. In order to determine the validity of this assumption, you w have to analyze the time constant of the circuit and make sure that it is reasonably faster than the sampling rate.

Let us now move on to derive the equations of motion for our system.

Equation of Motion

Before writing the equation of motion for the motor system, let me first list down some of the basic equations.

$$E_A = K\phi\dot{\theta}$$
$$\dot{E_A} = K\phi\ddot{\theta}$$
$$T = K\phi i$$

where:
E_A is the motor back-EMF
K is the motor machine constant
ϕ is the magnetic flux produced by the stator poles
$\dot{\theta}$ is the motor angular speed
T is the Torque of the motor
i is the current of the circuit

Now, let us begin with the mechanical system. Assuming that the motor is driving a propeller with rotational inertia of J and an air resistance of b which is proportional to the angular velocity of the propeller, we can derive the equation of motion as follows:

$$J\ddot{\theta} = T - b\dot{\theta} \text{ ———–- } (2)$$

where:

$\ddot{\theta}$ is the angular acceleration of the motor
J is the moment of inertia of the motor shaft and the propeller
b is the air resistance which is proportional to the angular velocity of the propeller

Then, from the circuit diagram in the above section of the motor alone without the extra resistor that we added, we can derive the following equation:

$$V = iR + E_A + L\frac{di}{dt}$$

Here, we are going to make a daring assumption that the dynamics of the circuit is much faster than the dynamics of the mechanical system thus $\frac{di}{dt} = 0$. As a result, we will get the following equation:

$$V = iR + E_A \text{ ———–- } (3)$$

There are ways to show that this assumption is valid but that is out of the scope of this tutorial. For those of you who are interested, you will have to determine the magnitude of the poles of the system in order to check if the above simplification is possible.

When you combine the equation for the mechanical system (2) and the electrical system (3), you will arrive at th following equation:

$$V = \frac{RJ}{k\phi}\ddot{\theta} + \frac{Rb}{k\phi}\dot{\theta} + E_A$$

Substituting in the basic equations given at the beginning of this section,

$$V = \frac{RJ}{(k\phi)^2}\dot{E_A} + \frac{Rb}{(k\phi)^2}E_A + E_A$$

$$\frac{(k\phi)^2}{RJ}V = \dot{E_A} + \frac{b}{J}E_A + \frac{(k\phi)^2}{RJ}E_A$$

Let $\frac{(k\phi)^2}{RJ} = \gamma$ and $\frac{b}{J} = \alpha$, then,

$$\gamma V = \dot{E_A} + \alpha E_A + \gamma E_A$$

Now, let us convert our simple differential equation into a difference equation.

$$\gamma V[n-1] = \frac{E_A[n] - E_A[n-1]}{t} + \alpha E_A[n-1] + \gamma E_A[n-1]$$

Here, $[n]$ refers to the time step n so $[n-1]$ refers to the value from the previous cycle. Simplifying the above equation,

$$E_A[n] = (1 - t\alpha - t\gamma)E_A[n-1] + t\gamma V[n-1] \text{ ------- } (4)$$

where t is the time interval between data points and $V[n-1]$ is our voltage input to the motor which can be controlled. The above equation is in red because it is one of the main equation that we will be working on.

Assuming a constant voltage input, the equation above is in the form:

$$E_A[n] = \lambda(E_A[n-1]) + C$$

This is a recursive equation which has a very simple solution. Let us first take a look at when $n = 1$

$$E_A[1] = \lambda(E_A[0]) + C$$

For the next n,

$$E_A[2] = \lambda(E_A[1]) + C$$

$$E_A[2] = \lambda^2(E_\lambda[0]) + \lambda C + C$$

In general,

$$E_A[n] = \lambda^n(E_A[0]) + \lambda^{n-1}C + \lambda^{n-2}C + \ldots + \lambda^1 C + \lambda^0 C$$

Notice that there is a geometric progression in the solution. When we simply this geometric progression, we w arrive at the following solution:

$$E_A[n] = \lambda^n(E_\lambda[0]) + \frac{C(1-\lambda^n)}{1-\lambda}$$

E_A is proportional to the motor angular velocity so a constant E_A refers to a constant motor speed while an oscillating E_A refers to an oscillating motor speed. From the above equation, when $|\lambda| < 1$, $\lambda^n < 1$ when $n \to \infty$. This in turn means that E_A will converge to a certain value after a period of time. However, if $|\lambda| > 1$, $\lambda^n \to \infty$ when $n \to \infty$ so the value E_A will get extremely big after some time.

Below is a summary of how the system response changes with the value of λ.

- If $\lambda > 1$, the sequence grows without bound thus the system response diverges
- If $\lambda = 1$, the sequence does not change value given that $C = 0$. Otherwise, the system response diverges
- If $0 < \lambda < 1$, the sequence converges to a specific value gradually
- If $-1 < \lambda < 0$, the sequence oscillates but converges to a specific value gradually
- If $\lambda = -1$, the sequence oscillates without ever converging
- If $\lambda < -1$, the sequence oscillates and the magnitude of their values grow without bound thus the system response diverges

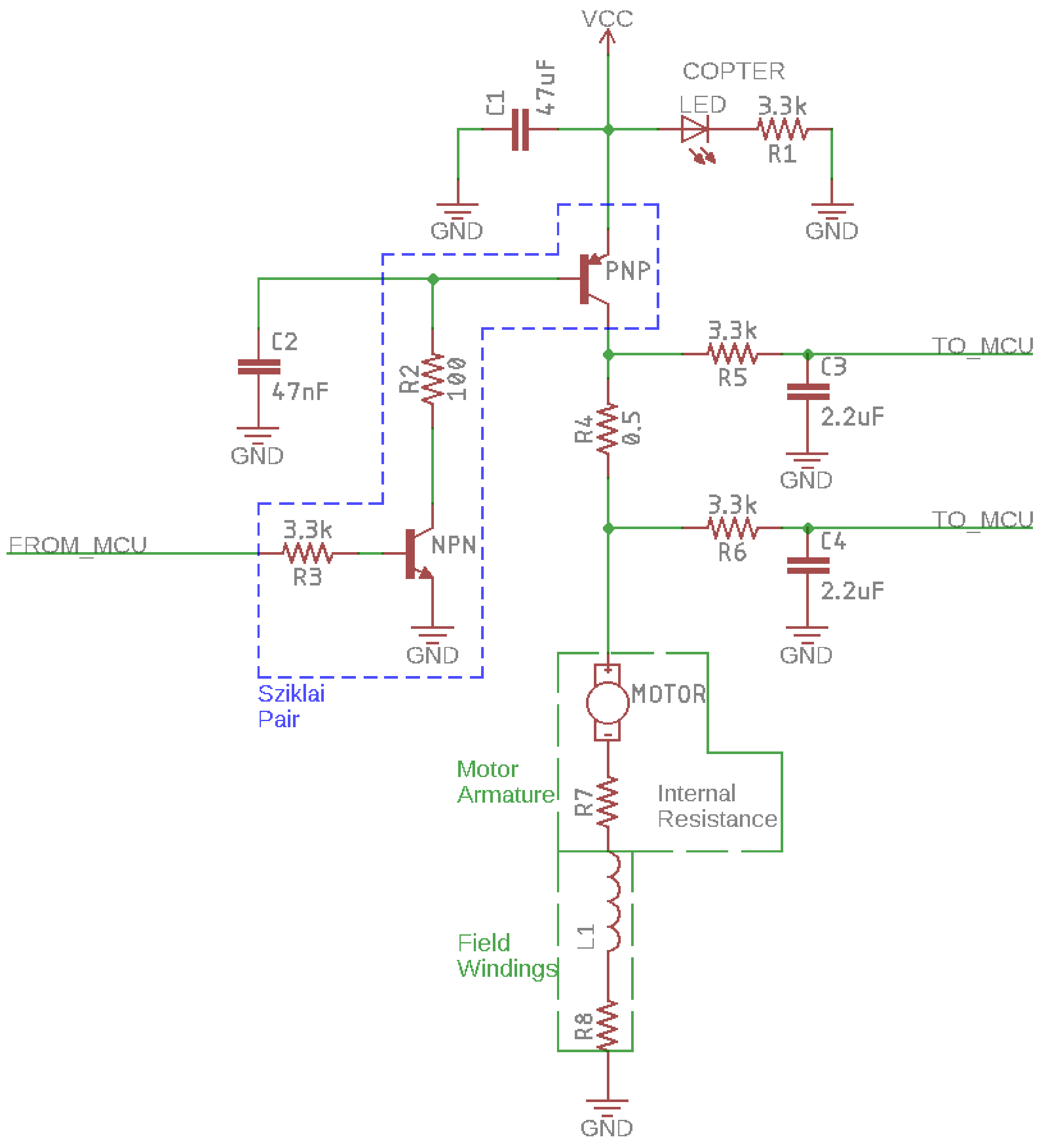
And this is it! We have determined the criteria for which the system can be considered stable or unstable.

Looking back at our main equation (equation 4 in red), λ in the previous equation is actually equals to the $(1 - t\alpha - t\gamma)$ term. In other words, if we can manipulate this term, we can control how the system responds to a change in input. In the "Proportional Control" section below, we will discuss about how we can manipulate the input so as to change the λ term as we see fit.

Calibration

In this section, we are going to determine several system constants that is required for controlling the response. From the previous section, you can already see that there are some constants that need to be determined, such as α and λ. The values given here in this section are all measured from my device so it will be different from yours. However, as long as you follow the steps here, the end result should be the same.

First, let us determine the internal resistance of the motor. This is needed in order to determine the back-EMF as shown by equation 1. When a motor spins, it will produce a back-EMF that will affect our measurements. The solution to this is simple. We just have to stop the motor from turning while doing our calibration of the internal resistor. If you have built the circuit as shown in the **previous chapter**, you can use the Arduino straight away for calibration (I have re-posted the diagram below for your easy reference). Otherwise, attach a resistor of a known value between the motor and the power source, then measure the voltage before and after the resistor just like the "Motor Circuit" diagram in this post.



CIRCUIT SCHEMATIC

From here on, I will assume that you are using the circuit shown in the **previous chapter**. Using the Arduino, set a certain PWM and attach the pin to the Sziklai Pair (shown as FROM_MCU). Then read the values using analogRead() for the 2 points shown as TO_MCU (labeled as V1 for the top and V2 for the bottom for the rest of this post) while the motor is forcefully stopped from turning. When the motor is not spinning, equation 1 reduces to:

$$R2 = V2/i$$

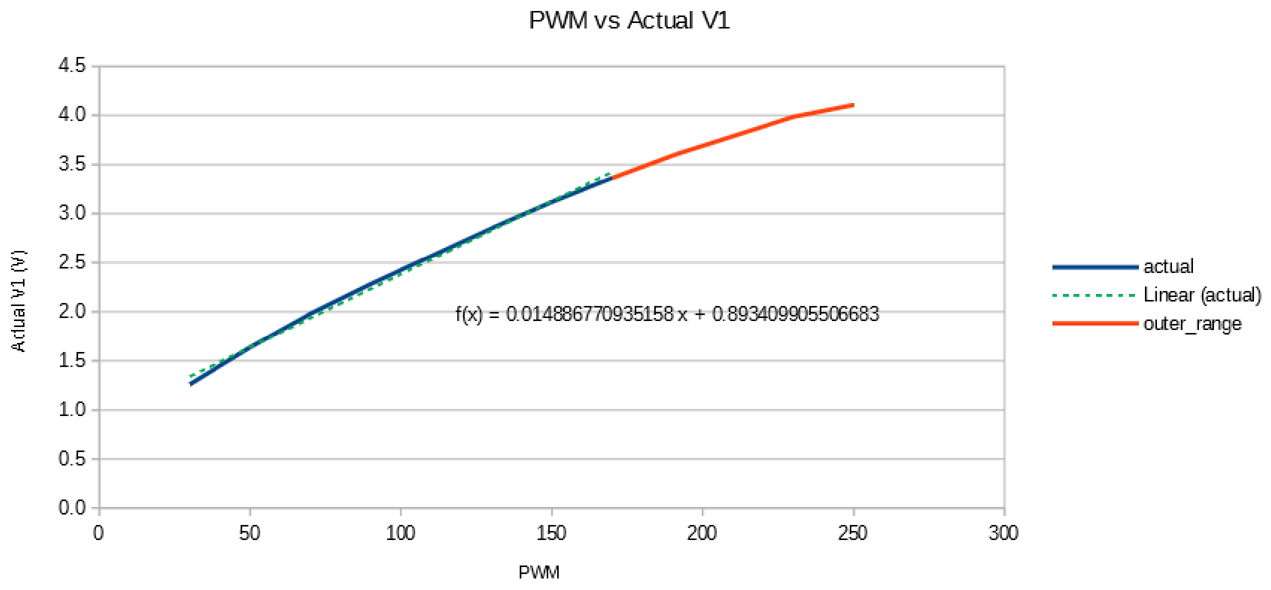
**I apologize for the confusion here. $R2$ in the equation is actually the internal resistance of the motor as shown in the “Motor Circuit” diagram, and this is equivalent to $R7 + R8$ in the “Circuit Schematic” diagram.

Below is a table of the values that I have measured.

PWM	V1	V2	PWM (V)	V1 (V)	V2 (V)	Current (A)	Motor Resistance (ohm)
			= PWM / 255 * 5	= V1 / 1023 * 5	= V2 / 1023 * 5	= (V1 – V2) / R4	= V2 / current
30	107	67	0.588235	0.522972	0.327468	0.391007	0.837500
50	155	96	0.980392	0.757576	0.469208	0.576735	0.813559
70	201	122	1.372549	0.982405	0.596285	0.772239	0.772152
90	255	158	1.764706	1.246334	0.772239	0.948192	0.814433
110	300	187	2.156863	1.466276	0.913978	1.104594	0.827434
130	350	221	2.549020	1.710655	1.080156	1.260997	0.856589
150	390	250	2.941176	1.906158	1.221896	1.368524	0.892857
170	435	275	3.333333	2.126100	1.344086	1.564027	0.859375
190	480	300	3.725490	2.346041	1.466276	1.759531	0.833333
210	515	325	4.117647	2.517107	1.588465	1.857283	0.855263
230	540	335	4.509804	2.639296	1.637341	2.003910	0.817073
250	565	350	4.901961	2.761486	1.710655	2.101662	0.813953
						AVERAGE	0.832794

Using this method, I have estimated that the internal resistance of the motor is 0.832794 Ohms.

Another point to take note here is that the PWM value does not correlate to the voltage input to the motor perfectly. This is because of the voltage drop across the transistor that is in use. I have plotted the values of PWM against V1(V) below and it should be obvious that it not directly proportional. PWM=0 represents V1=0, but PWM=255 does not mean V1=5V.



PWM VS ACTUAL VOLTAGE(V1)

I linearized the section from PWM=30 to PWM=170 and the curve fitting went quite well. Now, we have an equation to estimate the relationship between PWM and actual motor voltage input as shown by the equation below. However, we must take note that this estimate will be erroneous for values larger than PWM=170.

$$V[n-1] = (M)(I_{pwm}[n-1]) + C \text{ ------- (5)}$$

where $I_{pwm}[n-1]$ is the PWM input at time step $n-1$, $M = 0.01488677$ and $C = 0.8934099$.

Next up, I ran the tests again with the propeller spinning and I received the values as follows:

PWM	V1	V2	PWM (V)	V1 (V)	V2 (V)	Current (A)	Motor Resistance (ohm)	Back EMF (V)
30	258	218	0.588235	1.260997	1.065494	0.391007	0.832794	0.739865680050403
50	335	280	0.980392	1.637341	1.368524	0.537634	0.832794	0.92078549579755
70	405	332	1.372549	1.979472	1.622678	0.713587	0.832794	1.02840735876061
90	467	380	1.764706	2.282502	1.857283	0.850440	0.832794	1.14904167620151
110	525	421	2.156863	2.565982	2.057674	1.016618	0.832794	1.21104079745655
130	583	468	2.549020	2.849462	2.287390	1.124145	0.832794	1.35120962682134
150	638	509	2.941176	3.118280	2.487781	1.260997	0.832794	1.43763084553302
170	688	545	3.333333	3.362659	2.663734	1.397849	0.832794	1.49961413658097
190	735	580	3.725490	3.592375	2.834800	1.515152	0.832794	1.57299124040058
210	775	608	4.117647	3.787879	2.971652	1.632454	0.832794	1.612155245[illegible]097
230	815	635	4.509804	3.983382	3.103617	1.759531	0.832794	1.6382909658964

250	840	653	4.901961	4.105572	3.191593	1.827957	0.832794	1.66928261142038

Now, we have an idea of the range of the back-EMF of the motor! This enables us to specify the set point (desired speed) for the motor. I mean, you can set the desired back-EMF (equivalent to setting the speed because both parameters are proportional) to be 3V, but the system will never be able to achieve that. That is why we are doing this calibration. For my case, I capped the input setPoint to 1.65V and broke it up into 255 intervals. This means that an input of 0 is equivalent to 0V, and an input of 255 is equivalent to 1.65V.

Since we now have an equation relating the PWM and the voltage input to the motor (equation 5), let us substitute it into the equation of motion (equation 4 in red). This will yield the following equation:

$$E_A[n] = (1 - t\alpha - t\gamma)E_A[n-1] + t\gamma(M)(I_{pwm}[n-1]) + t\gamma C \text{ ------ (6)}$$

Next up, we have to determine the values of α and γ but we will need the proportional control to determine them so I'll talk about it in the "Proportional Control" section below instead.

Proportional Control

In the section above, I wrote that we can manipulate the system response by setting an appropriate input to the motor. That is what we are going to do in this section here, but before we begin, we need to determine the values of α and γ. Since there are 2 unknowns, we will require 2 equations to solve for their values.

Equation 1

Let us start by doing some analysis on the system at time ∞ (which actually just means after some period of time). Assuming that the system response converges,

$$E_A[n-1] = E_A[n] = E_A[\infty]$$

This just means that the value of E_A stops changing with time. Substituting this into equation 4, we have:

$$E_A[\infty] = (1 - t\alpha - t\gamma)E_A[\infty] + t\gamma V[\infty]$$

$V[\infty]$ is simply our input which is not going to change with time (at least until the system response converges) so I am just going to label it as V_{input}. Notice that this is actually the value of V1 in the "Motor Circuit" diagram. Simplifying the above equation, you will arrive at:

$$E_A[\infty] = \frac{\gamma}{\alpha+\gamma} V_{input} \text{ ------ (7)}$$

If our mathematical model is accurate, we can let

$$V_{input} = \frac{\alpha+\gamma}{\gamma} E_{A\_desired}$$

then substituting it back into equation 7,

$$E_A[\infty] = E_{A\_desired}$$

This means that we can control the final value of the back-EMF provided that we know the values of α and γ. In another sense, equation 7 is actually one of the 2 equations that is required to determine the values of α and γ.

Equation 2

In order to get the second equation to solve for the unknowns, we have to modify our input to the system a little. Assuming that our PWM input is as follows:

$$I_{pwm}[n-1] = K_p(E_{A\_desired} - E_A[n-1]) + 30$$

where K_p is a constant of proportionality that we can change freely. Substituting this into equation 6 (30 is actually just a random value. It can be anything),

$$E_A[n] = (1 - t\alpha - t\gamma)E_A[n-1] + t\gamma(M)(K_p(E_{A\_desired} - E_A[n-1]) + 30) + t\gamma C$$

Simplifying and grouping all the constants, we will get:

$$E_A[n] = (1 - t\alpha - t\gamma - t\gamma M K_p)E_A[n-1] + Constant$$

Notice that this is still in the same form as

$$E_A[n] = \lambda(E_A[n-1]) + C$$

but now, the system's natural frequency λ is given by:

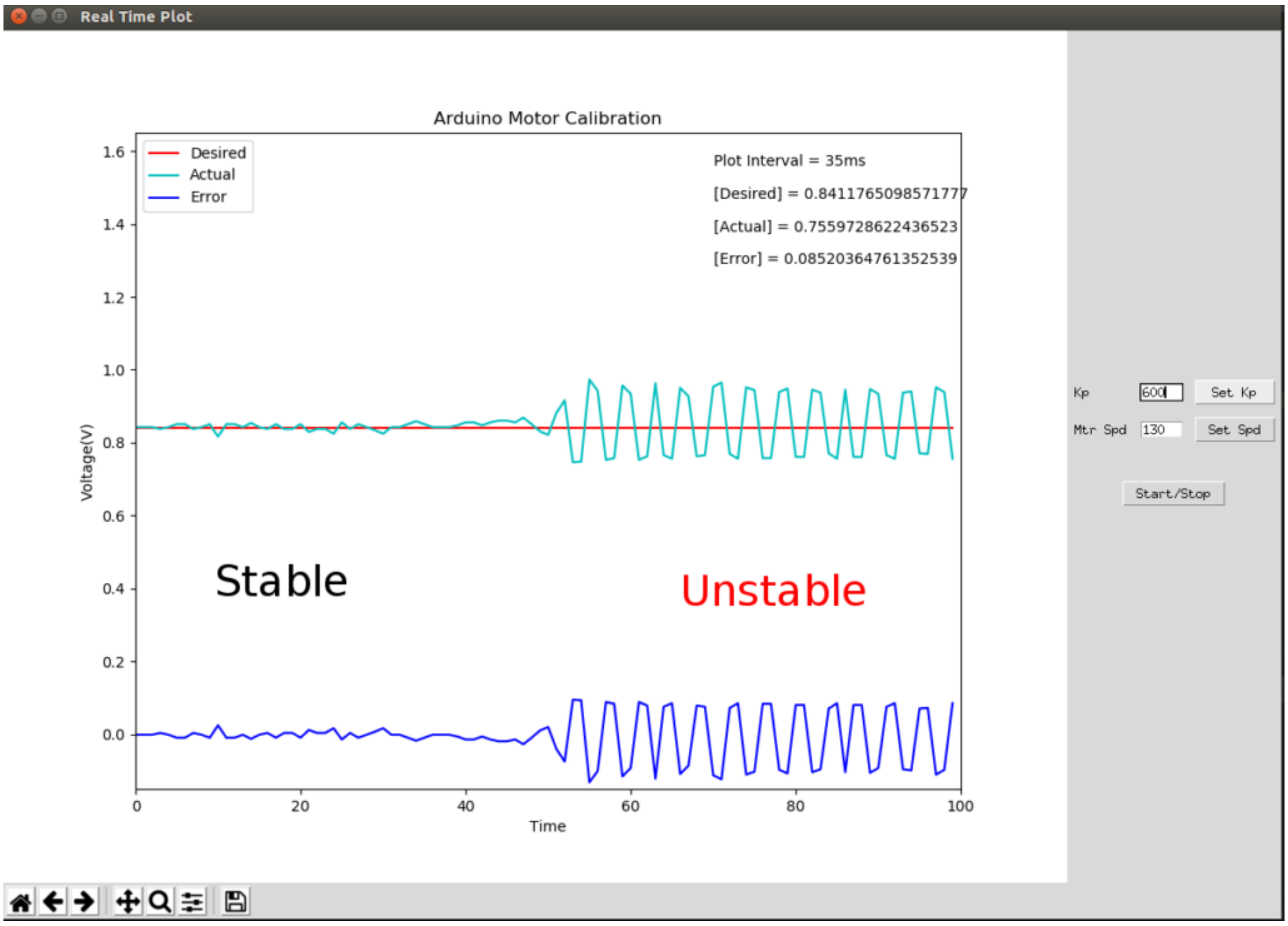
$$\lambda = 1 - t\alpha - t\gamma - t\gamma M K_p \text{ -------- (8)}$$

K_p is our parameter so it means that we can manipulate the system's natural frequency as we see fit! What we are going to do now is tune the value of K_p such that $\lambda = -1$. When $\lambda = -1$, the system oscillates indefinitely without growing unstable theoretically. Practically, external disturbances and inaccurate modelling prevents this special case from happening. Despite this, it is still possible to tune the system till the response gradually becomes unstable and estimate the K_p value for $\lambda = -1$.

I will give an example of 1 trial here where the constant is set to be 30. First, let the motor command (PWM) be:

```
int motorCmd = int(g_Kp * error + 30.0);
motorCmd = min(max(motorCmd, 0), 255);
```

where "g_Kp" is our K_p and "error" is the difference between the desired back-EMF and the actual back-EMF. We will run our system with $K_p = 0$ first to check the back-EMF when the constant is 30. For my case, when the constant is 30, the error is roughly 0 when the desired back-EMF is set to 130 (0V~1.65V is converted to 0~255 for my scale). This is approximately equals to a back-EMF of 0.841176V (130/255*1.65). Next, increase the value of K_p gradually such that the system response turns unstable. Below is a screenshot of what you should see when the system is unstable.



Plot Interval = 35ms
[Desired] = 0.8411765098571777
[Actual] = 0.7559728622436523
[Error] = 0.08520364761352539

Kp 600
Mtr Spd 130

MOTOR RESPONSE

Here are my results:

PWM	E_A	K_p
30	130	600
40	150	610
50	165	580
60	182	550
70	195	570

PWM column represents the constant that is given to the motor (30 in the example case). E_A column is the corresponding motor speed to the PWM input in our own custom scale. And lastly, the K_p column is the value at which the system turns unstable. We are now ready to solve for α and γ

Determining α and γ

Combining equation 7 and 8, we will get the system of linear equation as shown below. Note that I used a time interval of 20ms so if yours is different, you cannot use the same values as written in the equation below:

$$\begin{bmatrix} E_A[\infty] & E_A[\infty]-V_{input} \\ 1 & 1+MK_p \end{bmatrix}\begin{bmatrix} \alpha \\ \gamma \end{bmatrix}=\begin{bmatrix} 0 \\ 100 \end{bmatrix}$$

CALIBRATION EQUATION

This equation is in the form of

$AX = B$

where all the parameters are matrices. You can solve this by multiplying the inverse of A with B. I have written a spread sheet using LibreOffice to make this simpler so you can use it if you want. You can download it **here** but take note that it is not an excel spread sheet (too poor to purchase Microsoft Office! T.T). LibreOffice is free btw 🙂

Anyway, we now have all the values of the unknowns. For me, that is

$M = 0.014887$
$C = 0.893410$
$\gamma = 9.803448$
$\alpha = 5.359968$

Now, let us begin with the real Proportional Control.

Proportional Control Implementation

I am going to start by giving you the answer. This is not the perfect answer though, as I am sure that there are better solutions out there too. Here is the equation of motion once again (equation 6):

$$E_A[n] = (1 - t\alpha - t\gamma)E_A[n-1] + t\gamma(M)(I_{pwm}[n-1]) + t\gamma C$$

Let us implement our voltage input to the motor as follows:

$$I_{pwm}[n-1] = \frac{K_p}{M}(E_{A\_desired}[n-1] - E_A[n-1]) + \frac{\alpha}{\gamma M}E_{A\_desired}[n-1] + E_{A\_desired}[n-1] - \frac{C}{M}$$

Here, the term with K_p is our proportional controller. You can see now that It is called the proportional controller because we are scaling the error, the difference between the desired output and the actual output $(E_{A\_desired}[n-1] - E_A[n-1])$ by a specified constant of proportionality (K_p) before feeding it back to the system.

The overall system response will then be:

$$E_A[n] = (1 - t\alpha - t\gamma - t\gamma K_p)E_A[n-1] + (t\alpha + t\gamma + t\gamma K_p)E_{A\_desired}[n-1]$$

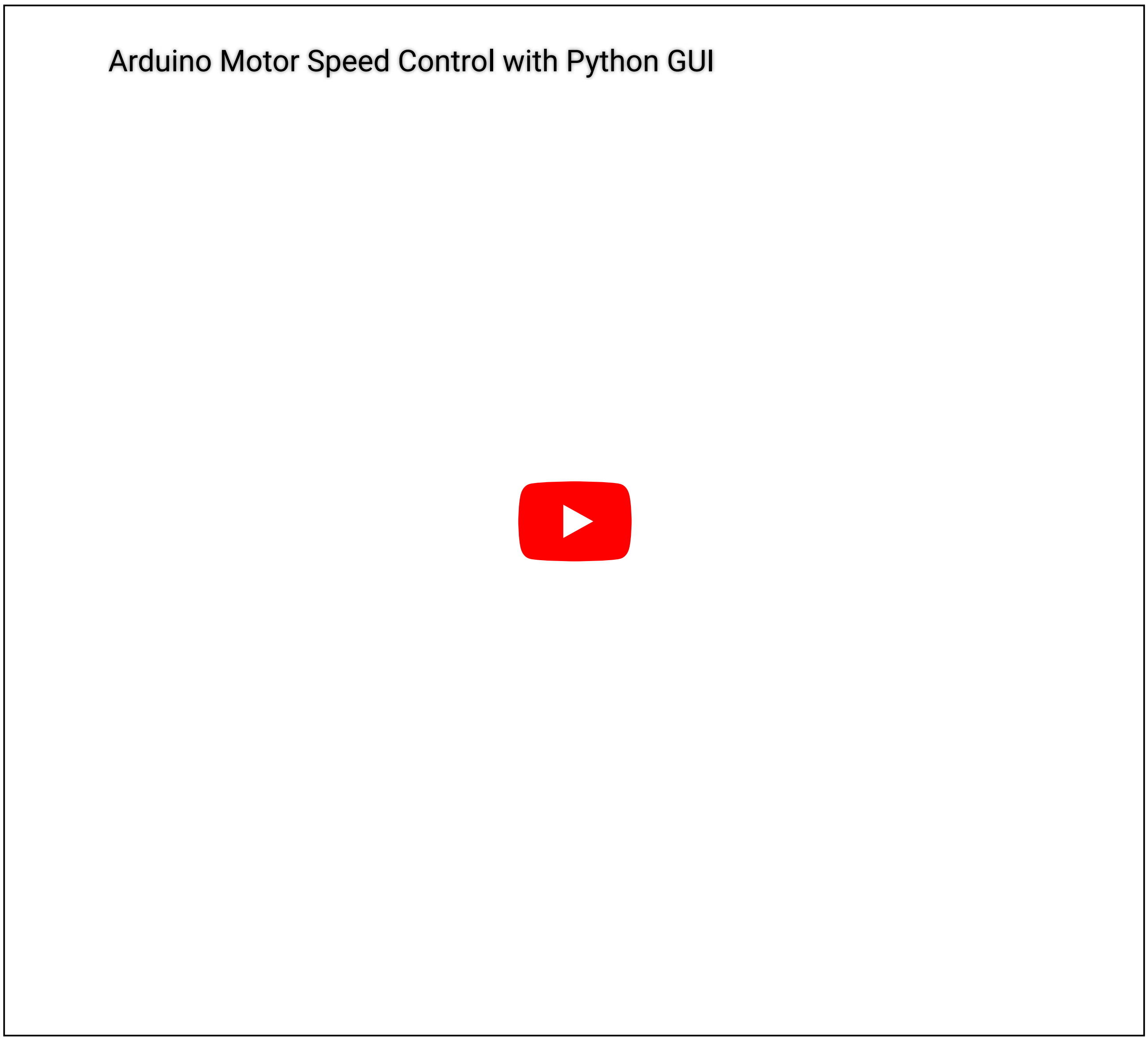
Assuming we choose a value of K_p such that $|(1 - t\alpha - t\gamma - t\gamma K_p)| < 1$, the system will be stable. Thus,

$$E_A[n-1] = E_A[n] = E_A[\infty]$$

$$E_A[\infty] = (1 - t\alpha - t\gamma - t\gamma K_p)E_A[\infty] + (t\alpha + t\gamma + t\gamma K_p)E_{A\_desired}[\infty]$$

$$E_A[\infty] = E_{A\_desired}[\infty]$$

Assuming out mathematical model is correct, the back-EMF of the motor will always converge to the value that we desire it to be. But is this true in practise? Yes and No. It will be true in the region where our approximation is correct (remember our linearization of the PWM input to the actual voltage?), and once you start entering values that are away from this region, the system will not give accurate results. There are ways to mitigate this such as adding a derivative control or an integral control but let's leave that for the next chapter. For now, great work on finishing till this point! Below is a video of the system response captured with the **Python code** that I wrote.



On a final note, the voltage input to the motor can be anything that you want it to be. Can you come up with a better controller for this? Feel free to post any suggestions to improve the response! 🙂

Go to Next Section

PREVIOUS POST
Creating a Graphic User Interface (GUI) with Python

NEXT PO
Copter Angle Control (Relative)

POSTED IN FEEDBACK CONTROL SYSTEM

7 comments on "*Motor Speed Control*"

Lukas
June 6, 2019 at 12:58 am

How did you get such "constant" values for V1 and V2 (in both cases, fixed and not fixed). In my case measuring these voltages results in small jumps.
Is it because the low-pass filtering does not work properly? The problem might be the capacitors, hmm…
Would be great if someone was able to give me a hint.
Best regards,
Lukas

rikisenia.L
June 6, 2019 at 8:45 pm

Hi Lukas,

The values of V1 and V2 were fluctuating as well for me.
I just took what seems like the average value during the fluctuation as my measurement.
We just need to have something close to the "correct" value and the feedback controller will do the rest of the work for us.

Lukas
June 11, 2019 at 4:06 am

Did you take the average of let's say 20 or 50 samples? Or even more fancy things like weighting standard deviations or really only looking "what could be a mean value" by plotting the analogs? I mean I often get peaks in my mesurements (e.g. normal value range 625-635 and than a sudden 595.
Combining this with my not soldered breadboard configuration my motor resistance is anywhere between 0.6 Ohm and 2.xx Ohm.
Can't proceed like that.

Best regards,
Lukas

rikisenia.L
June 11, 2019 at 10:55 pm

Hi Lukas,

I did not take any average values but simply just pick a value that appeared to be the mean while the readings are fluctuating. By right, if we want to do this calibration properly, we should do like what you said; collect the data over a period of time and take the average of them. However, since the control system should attenuate effects of the inaccuracies of our models, I did not spend that much effort into getting the perfect value for the model.
If you consider the sudden peaks in your measurement to be outliers and ignore them, do you still get huge discrepancy between the readings at different configurations?

Lukas
June 19, 2019 at 5:22 am

Quality of my measurement is better now after soldering everything together – breadboard connection was definitely not sufficient.
Can you tell my why your desired back-EMF for PWM 30 is
0.841176V and not 0.739865680050403V from the table?

rikisenia.L
June 19, 2019 at 9:46 pm

Hi Lukas,

This was more than a year ago so I cant remember the exact details but I think I was actually altering the [Mtr Spd] input to get an error that is close to 0.
When the constant in the code is set to 30, a value of 130 for the [Mtr Spd] gave an error close to 0 so I simply used 130 as a reference point.
In fact value 0.841176 actually came from (130/255)*1.65, where 1.65 is the upper limit of the back EMF that I have set.

I apologize for the confusion.
I'll change the writing there.

Lukas
June 25, 2019 at 5:51 am

Riki,
thanks for your explanation (&update)!
Still, I'm impressed about how "stable" your back-EMF behaves in the plot.
Your "[Actual]" – is it simply based on E_A=V_2 – R_motor*(V_1-V_2)/0.5Ohm ?
with V_2 and V_1 RC-filtered?
When I do this my B-EMF values rely on the arduino measurement for V_1 and V_2. Considering your case "PWM 130 motor spinning": V_2 = 2.287390V.
My mean value for V_2 is close to it but my oscilloscope shows the value jumps between 2.0 and 2.7 V due to the ~490Hz PWM which is obviously not "flat" enough after the RC-Circuit. Of course this kills my E_A – equation from above as the analog read gets both – the 2.0 V and the 2.7V resulting in oscillating E_A.
Was V_2 "DC-enough" to simply calculate E_A like this?
Best regards,
Lukas

Comments are closed.

Categories

Select Category

Archives